TiGER: Tiny bandwidth key encapsulation mechanism for easy miGration based on RLWE(R). ⋆

Seunghwan Park, Chi-Gon Jung, Aesun Park,
Joongeun Choi, and Honggoo Kang

Defense Counter-intelligence Command
horriblepaper@gmail.com, wjdclrhs@gmail.com
aesunpark18@gmail.com, joongeuntom@gmail.com
honggoonin@gmail.com

Abstract. The quantum resistance Key Encapsulation Mechanism (PQC -KEM) design aims to replace cryptography in legacy security protocols. It would be nice if PQC-KEM were faster and lighter than ECDH or DH for easy migration to legacy security protocols. However, it seems impossible due to the temperament of the secure underlying problems in a quantum environment. Therefore, it makes reason to determine the threshold of the scheme by analyzing the maximum bandwidth the legacy security protocol can adapt. We specified the bandwidth threshold at 1,244 bytes based on IKEv2 (RFC7296), a security protocol with strict constraints on payload size in the initial exchange for secret key sharing. We propose TiGER that is an IND-CCA secure KEM based on RLWE(R). TiGER has a ciphertext (1,152bytes) and a public key (928bytes) smaller than 1,244 bytes, even at the AES256 security level. To our knowledge, TiGER is the only scheme with such an achievement. Also, TiGER satisfies security levels 1, 3, and 5 of NIST competition. Based on reference implementation, TiGER is 1.7-2.6x faster than Kyber and 2.2-4.4x faster than LAC.

Keywords: Post quantum cryptography migration · Ring learning with error (RLWE) · Ring learning with rounding (RLWR) · Key encapsulation mechanism (KEM)

1 Introduction

PQC-KEM (Post Quantum Cryptography Key Encapsulation Mechanism) must be easily migrated to legacy security protocols ((D)TLS, IKE, SSH, IPSEC, etc.), and performance, device memory, and communication bandwidth must be similar to current KEM such as ECDH and DH for migration.

Analyzing the algorithms submitted to the NIST competition, it seems reasonable to choose the Lattice problem for migration. The KEM designed based

⋆ This work is submitted to 'Korean Post-Quantum Cryptography Competition' (www.kpqc.or.kr).

on LWE, and its variants (Ring/Module LWE(R)) is fast, so it does not constrain migration. However, KEM based on LWE and its variants are unsuitable for the resource constraint device and communication bandwidth because the size of the public key, private key, and ciphertext are more significant than that of DH or ECDH, which are legacy KEM. In this respect, The NIST competition also used the size of the public key and the size of the ciphertext as principal evaluation factors. Intuitively, structured RLWE(R) can have smaller bandwidth and efficient performance than less structured MLWE(R). Nevertheless, the bandwidth and operation speed of the RLWE(R) algorithm submitted to the NIST competition were comparable to those of MLWE(R)-based Kyber or Saber, so they were all rejected. However, we have the insight that NIST's first chosen KEM standard, Kyber, is a bit unsuitable to use for legacy security protocols. The bandwidth of Ring/Module LWE(R) cannot be as small as DH and ECDH, but migration is much easier when the security protocol fragmentation can be avoided. According to RFC 7296, IKEv2 implementations must be able to send, receive, and process IKE messages that are up to 1,280byte. And the first exchange message, IKE-SA-INIT, cannot be fragmented (RFC 7383). So it is necessary to transmit the ciphertext or public key for shared key generation in a payload of about 1,244 bytes, excluding the header. The Round5 team [8] also analyzed that ciphertext and public key sizes of about 1,300 bytes are required in an Ethernet. If we want to share an encryption key for AES256, which is believed to be secure even in a quantum environment, we should use Kyber1024. Since the public key and ciphertext size in Kyber1024 are 1,568 bytes each, complex implementations may be required for use with IKEv2.

We propose TiGER a compact IND-CCA secure KEM based on RLWR and RLWE where the public key and the ciphertext size do not exceed 1,244 bytes on all parameters. TiGER generates a public key based on RLWR, generates a ciphertext based on RLWE, and compresses it to achieve small bandwidth. RLWE(R) are structured lattices, but specific attacks that apply only to these are unknown and have been well-studied for a relatively long time. Also, we choose the RLWE(R) modulus $q = 256$ to cut-off a bandwidth. Since the hardness of RLWE(R) is determined by the dimension n and the σ/q, where σ is the standard deviation of noise distribution, choosing a small q is not a harmful hardness if determining a proper noise distribution [27,21]. If an error is added to a small q and even ciphertext compression is performed, the decryption failure rate will be very high. We solve this problem with error correction codes `XEf` [8] and `D2` [6]. In particular, the unique characteristics of the ring $f(X)=X^n+1$ in $R_q := \mathbb{Z}_q[X]/(f(X))$ are a good combination with the error correction code. That is, in the $X^n + 1$, n is constrained to the power of 2, and since it is sufficiently more significant than the required message size (128, 192, 256 bits), this buffer is used as a valuable space for redundancy bits. We use a sparse ternary uniform distribution with hamming weights for the secret/error distribution. It chose to enjoy being able to fine-tune the standard deviation through hamming weights while minimizing the propagation of errors. As constraints n and fixed q, parameters that are challenging to handle can be supplemented with hamming weights. It also allows to replace polynomial multiplication with bit-wise operations.

As a result, TiGER is the only one among all LWE-based KEMs that competed in the NIST competition; the size of the ciphertext and public key is smaller than 1,244 bytes each, so it can easily migrate to the legacy security protocol. In addition, TiGER128, TiGER192, and TiGER256 satisfy NIST competition security levels 1, 3, and 5, respectively, according to latest security strength estimation in MATZOV [24,5] and May [25]. It is more fine-grained, state-of-arts security strength estimates than Kyber [7] or Saber [13].

1.1 Design rationale

TiGER is an IND-CCA-secure Key Encapsulation Mechanism (KEM) based on the hardness of solving the learning-with-rounding and learning-with-errors problem over Ring lattices (RLWR and RLWE problem). The TiGER scheme is constructed in two-steps: we first introduce and IND-CPA-secure PKE(Public Key Encryption) scheme encrypting messages of a length of 16, 32 bytes. We then use a Fujisaki–Okamoto (FO) transform by Jiang et al. [17] to construct the IND-CCA-secure KEM.

Choice of the Ring We use $f(X)=X^n + 1$ in $R_q := \mathbb{Z}_q[X]/(f(X))$, where n is the power of 2. It is the common choice used by most of the Ring-lattice submitted to NIST competitions. The common ring has the advantage in that the polynomial modular reduction operation is straightforward, and there have been no known attacks exploit it [20]. On the other side of this choice, $f(X)=X^{n+1}+1$ in [8], where $n+1$ is prime and $f(X)=X^n - X^{n/2}+1$ in [26], where $n = 2^a 3^b$(a and b are positive integer) have the flexibility to select the appropriate degree n for each security level. Although we observed these recent studies meaningfully, we decided to enjoy the aspect of conservative security that there was no particular weakness even though $f(X)=X^n+1$ was well studied and the attribute of making the probability of decoding error the smallest. In addition, the constraint of degree n is tricky to witness as a disadvantage because it provides a valuable buffer space in terms of employing the error correcting code.

Choice of modulus All integer modulus in the scheme are power of 2. It improve performance by replacing $\lfloor (p/q) \cdot \mathbf{x} \rceil$ with ADD and AND operations [11]. Especially, fixed $q = 256$ is a byte size. The first proposed the choice of modulus q below the byte size with $q = 251$ in `LAC` [21], which won the PQC competition in china. In addition, `LizarMong` [18] designed the scheme with $q = 256$, which is the power of 2 like ours, and then LAC also added $q = 256$ parameter as an option during the NIST competition. Intuitively, this choice enjoys a small bandwidth and improved performance. It also provides very efficient modulo operation and memory usage and is suitable for single instruction multiple data (SIMD) implementations such as AVX2 and NEON. Even though the modulus is small, it can not affect the security since we maintain the error rate by selecting proper error distribution [21,27]. The modulus p used for RLWR and the modulus k_1, k_2 used for ciphertext compression are also the power of two.

Distribution We use a sparse ternary secret with a hamming weight [11] and [8] proved the hardness of the sparse ternary secret variants RLWE(R). Multiplication of sparse ternary secret polynomials can be replaced with bit operation to improve performance [2,20]. It also maintains correctness by preventing decryption errors from increasing and can select the optimized standard deviation for each security level by finely adjusting the hamming weight and has the advantage of having resistance to high hamming weight attacks [21] that manipulate the centered binomial distribution.

Adopt error correction code $R_q := Z_q[X]/X^n+1$ constrains n to the power of 2, so choose $n = 512$ or $n = 1024$ depending on the security level. Since general RLWE(R) schemes map one bit message to one coefficient in R_q, n is larger than the required size of the shared key (128, 192, 256 bits). We claim that RLWE(R) has the best compatibility with error correcting code (ECC) regarding message processing. A redundancy bit is inevitably required for error correction. n larger than the length of the shared key can be used as a buffer sufficient to use the redundancy bit. In particular, Because the RLWE(R) is a variant to maximize performance and reduce the size(ciphertext, public key) of the LWE, RLWE(R) with comparable performance and size to less structured MLWE is meaningless [4]. Thus, a small q selection is needed in RLWE(R), increasing the decoding failure probability. So, the combination of RLWE(R) and ECC is beautiful because ECC can be an excellent choice to overcome these drawbacks. We use the well-studied `XEf` [8] and `D2` [6] to utilize the buffer space. Since `XEf` avoids table look-up and branch conditions, it resists timing attacks [8]. That is, a 256-bit message is encoded with `XE5` to make a 512-bit code word (234 redundancy bits and 22 padding bits), and `D2` encodes the code word to make a 1024-bit $\hat{\mathbf{M}}$. Decoding is in reverse order.

Compress Public-key and Ciphertext NIST's candidate algorithms commonly use compression techniques. Public-key compression means sending only the *Seed* instead of $\mathbf{a}$ in R_q, and the receiver recovers $\mathbf{a}$ using the hash function. This reduces the public-key size from $2n \log q$ to size-of-$Seed$ + $n \log q$. Ciphertext compression is similar to the RLWR idea of discarding a few LSBs in $\mathbf{c_1}$, $\mathbf{c_2}$. IND-CCA KEM also can do the same. Ciphertext compression affects the security strength and decryption failure rate of the scheme. See subsection 5.4 and subsection 3.5 for an analysis.

1.2 Advantages and limitations

1.2.1 Advantages

- **Compact :** TiGER has the smallest[1] ciphertext and public key size among the LWE (include variants) algorithms that round2 of the NIST competition. That is a natural result since we chose a small coefficient $q = 256$, generated the public key RLWR, and compressed the ciphertext that made by RLWE.

[1] Evaluating the IND-CCA secure scheme with security level 5

- **Easily migration :** Our goal in designing PQC-KEM is to replace key exchange algorithms in security protocols such as (D)TLS, IKE, SSH, IPSEC, and DNSSEC. We claim that if the public key or ciphertext size is larger than 1,244 bytes, migration to existing security protocols may be restricted. We strongly agree with the Round5 team's insight [8]. In other words, Ethernet MTU is commonly 1,500 Bytes, but the payload, excluding the header size, is 1,364 bytes (1,364 Bytes payload also must include important information, such as cookies, that can be used to determine whether or not a DoS attack 12-56 bytes). After all, the bandwidth we can send ciphertext or public key is about 1,300 bytes. We explored more stringent payload size constraints in IKEv2, a security protocol required for key exchange. IKEv2 (RFC 7296), a security protocol with strict constraints on payload size in the initial exchange for secret key sharing, implementations must be able to send, receive, and process IKE messages that are up to 1,280 bytes. And the first exchange message, IKE-SA-INIT, cannot be fragmented (RFC 7383). So it is necessary to transmit the ciphertext or public key for shared key generation in a payload of about 1,244 bytes, excluding the header. So, migration is much more advantageous if the public key and ciphertext size are smaller than 1,244 bytes. To our knowledge, TiGER is the unique IND-CCA KEM with a ciphertext and public key size smaller than 1,244 bytes at the AES256 security level.
- **High performance :** TiGER samples the secret $\mathbf{s}$ and the noise $\mathbf{e}$ from a sparse ternary uniform distribution, making polynomial multiplication simple and efficient. Furthermore, since all moduli are a power of 2, we do not require explicit modular reduction.
- **Friendly for SIMD :** TiGER is friendly for SIMD, such as AVX2 and NEON, due to the modulus q=256 (8 bits). For example, C intrinsic data types `__m256i` can be stored in a 32-dimensional 8-bit integer so that only 16 `__m256i` data types can express TiGER parameters.

1.2.2 Limitations

- Side-channel analysis attack surface is larger than Non-ECC. However, we believe that side-channel attacks will be solved with time and the wisdom of crowds(as in the case of RSA and ECC). The `XEf` resists timing attacks[30]. Some attacks assume more powerful attackers, but we firmly believe that the respected cryptography community will solve them.
- Although RLWE and RLWR have proven hardness and have been studied for longer than MLWE(R), their security is questioned because they are structured lattices. However, there are no known attacks in RLWE and RLWR, and RLWE is well-studied because of the homomorphic encryption, so there is no doubt about it.

2 Preliminaries

2.1 Public Key Encryption

A public key encryption (PKE) scheme is a cryptographic system that uses a pair of a public key and a corresponding private key. The security of PKE depends on maintaining the confidentiality of the private key. The public key can be publicly distributed, and anyone can encrypt using the public key that only the user who has the private key can decrypt it. The following is the syntax of PKE.

Definition 1 (PKE). *A PKE scheme consist of three algorithms* ***KeyGen****,* ***Encryption****,* ***Decryption*** *which are defined as follows:*

- ***KeyGen***(1^λ)*: The key generation algorithm takes as input a security parameter* 1^λ*. It outputs a public key* ***pk*** *and a private key* ***sk****.*
- ***Encryption(pk, M)****: The encryption algorithm takes as input the public key* ***pk*** *and a message* $\boldsymbol{M} \in \mathcal{M}$*. It outputs a ciphertext* ***c****.*
- ***Decryption(sk, c)****: The decryption algorithm takes as input the private key* ***sk*** *and the ciphertext* ***c****. It outputs the message* ***M*** *or* $\perp$*.*

The correctness property of PKE is defined as follows: For all ***pk*** *and* ***sk*** *generated by* ***KeyGen***(1^λ)*,* ***c*** *generated by* ***Encryption(pk, M)*** *for* ***M****, it is required that*

- *If* ***sk*** *is valid, then* ***Decryption(sk, c)*** $=$ ***M****.*
- *If* ***sk*** *is invalid, then* ***Decryption(sk, c)*** $= \perp$*.*

2.2 Key Encapsulation Mechanism

A key encapsulation mechanism (KEM) is used to share the secret key of symmetric key encryption systems. The sender generates a ciphertext for sharing the secret key and sends it to the receiver. The receiver decapsulates the ciphertext and generates a secret key to be used for symmetric key encryption. The following is the syntax of KEM.

Definition 2 (KEM). *A KEM scheme consist of three algorithms* ***KeyGen****,* ***Encapsulation****,* ***Decapsulation*** *which are defined as follows:*

- ***KeyGen***(1^λ)*: The key generation algorithm takes as input a security parameter* 1^λ*. It outputs a public key* ***pk*** *and a private key* ***sk****.*
- ***Encapsulation(pk)****: The encapsulation algorithm takes as input the public key* ***pk****. It outputs a ciphertext* ***c*** *and a shared key* ***K****.*
- ***Decapsulation(sk, c)****: The decapsulaiton algorithm takes as input the private key* ***sk*** *and the ciphertext* ***c****. It outputs the shared key* ***K*** *or* $\perp$*.*

The correctness property of KEM is defined as follows: For all ***pk*** *and* ***sk*** *generated by* ***KeyGen***(1^λ)*,* ***c*** *and* ***K*** *generated by* ***Encapsulation(pk)****, it is required that*

- *If* ***sk*** *is valid, then* ***Decapsulation(sk, c)*** $= \hat{\boldsymbol{K}}$ *such that* $\boldsymbol{K} = \hat{\boldsymbol{K}}$*.*
- *If* ***sk*** *is invalid, then* ***Decapsulation(sk, c)*** $= \hat{\boldsymbol{K}}$ *or* $\perp$ *such that* $\boldsymbol{K} \neq \hat{\boldsymbol{K}}$*.*

2.3 Related Works

Lattice-based Public-Key Encryption. Due to the threat of quantum computers to the classical cryptography such as RSA or Diffie-Hellman, lattice-based cryptography is attracting attention as one of various post-quantum cryptography. The first lattice-based cryptographic construction was introduced by M. Ajtai [1]. And, the first lattice-based public key encryption (PKE) that called NTRU scheme is proposed by J. Hoffstein et al. [15]. The security of thier PKE scheme was not proven under worst-case hardness assumptions. In 2005, O. Regev [29] introduced the first lattice-based PKE scheme with the Learning With Errors (LWE) problem. The LWE problem is as difficult to solve as the worst-case lattice problems. Since then, a variety of lattice problems such as Learning With Rounding (LWR), Ring-LWE (RLWE), Ring-LWR (RLWR), etc. have been proposed [22,28,23,9], and these problems have been widely used to design public key cryptography. Also, after the process of standardizing post-quantum cryptography (PQC), many other lattice-based PKE schemes such as KYBER [7], NewHope [6], LAC [21], Round5 [8], Saber [13], and RLizard [20] assuming the hardness of lattice problems have been proposed.

CCA-secure KEM. Chosen-ciphertext attack (CCA) is a security model that allows an adversary to obtain the plaintext corresponding to the chosen ciphertext. Because the process of establishing a session key in communications is similar to accessing a decryption oracle by an adversary. It is very important to design the key encapsulation mechanism (KEM) considering the chosen-cipertext attack. Fujisaki and Okamoto [14] introduced a generic transformation method that can drive the chosen-cipertext secure KEM scheme from the chosen-plaintext secure PKE scheme. And, this method has been widely used to design many cryptographic algorithms. Recently, the Fujisaki-Okamoto transformation method in consideration of a quantum computing environment has been proposed [32,17,16]. Jiang et al. [17] presented Fujisaki-Okamoto transformation method that was proven tight security reductions in the quantum random oracle model.

3 Specification

3.1 Notation

In this subsection, we introduce some notations used in this document.
Let $\mathbb{Z}$ be the ring of rational integers. We define for an $x \in \mathbb{R}$ the rounding function $\lfloor x \rceil$, where $\lfloor x \rceil$ means the nearest integer to the x. We denote $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ which means the ring of integer polynomial modulo $(X^n + 1)$, where each coefficient is reduced modulo q.
In this paper, n is a positive integer expressed as a power of two, which means the dimension of RLWE samples. q and p mean modulus for RLWE and RLWR, respectively. We also use the modulus k_i, where $i = 1, 2$, for ciphertext compression. We use p, q, k_1, and k_2 which are all the power of 2.

Bold lower-case letters represent polynomials with coefficients in R_q. Multiplication in R_q is represented by $*$. $\lfloor \mathbf{a} \rceil$ is the rounding to the nearest integer for each coefficient in the polynomial $\mathbf{a}$. $x \parallel y$ is the concatenation of x and y. $HWT_n(h, Seed)$ is the uniform distribution over the subset of $\{-1,0,1\}^n$ whose elements contain $n-h$ number of zeros, and is generated using $Seed$. $\texttt{SHAKE256}(m, len)$ is a hash function that receives m and outputs a byte-string of the length len. eccENC and eccDEC are functions for encoding and decoding using the error correction codes.

3.2 Specification of TiGER.CPAPKE

Algorithm 1 `IND-CPA.KeyGen`

Input: The set of public *parameters*
Output: Public key $pk = (Seed_a \parallel \mathbf{b})$, Private Key $sk = (\mathbf{s})$

1: $Seed_a \xleftarrow{\$} \{0,1\}^{256}$
2: $Seed_s \xleftarrow{\$} \{0,1\}^{256}$
3: $\mathbf{a} \leftarrow \texttt{SHAKE256}(Seed_a, n/8)$
4: $\mathbf{s} \leftarrow HWT_n(h_s, Seed_s)$
5: $\mathbf{b} \leftarrow \lfloor (p/q) \cdot \mathbf{a} * \mathbf{s} \rceil$
6: $pk \leftarrow (Seed_a \parallel \mathbf{b})$ and $sk \leftarrow \mathbf{s}$
7: **return** pk, sk

Algorithm 2 `IND-CPA.Encryption`

Input: pk, Message $\mathbf{M} \in \{0,1\}^d$; Coin $w \xleftarrow{\$} \{0,1\}^{256}$
Output: Ciphertext $\mathbf{c} = (\mathbf{c_1} \parallel \mathbf{c_2})$

1: $\mathbf{r} \leftarrow HWT_n(h_r, w)$
2: $Seed_{e_1} \leftarrow (w + Nonce)$
3: $Seed_{e_2} \leftarrow (w + Nonce + 1)$
4: $\mathbf{e_1} \leftarrow HWT_n(h_e, Seed_{e_1})$ and $\mathbf{e_2} \leftarrow HWT_n(h_e, Seed_{e_2})$
5: $Seed_a, \mathbf{b} \leftarrow$ Parsing(pk)
6: $\mathbf{a} \leftarrow \texttt{SHAKE256}(Seed_a, n/8)$
7: $\mathbf{c_1} \leftarrow \lfloor (k_1/q) \cdot (\mathbf{a} * \mathbf{r} + \mathbf{e_1}) \rceil$
8: $\mathbf{c_2} \leftarrow \lfloor (k_2/q) \cdot ((q/2) \cdot \text{eccENC}(\mathbf{M}) + ((q/p) \cdot \mathbf{b}) * \mathbf{r} + \mathbf{e_2}) \rceil$
9: $\mathbf{c} \leftarrow (\mathbf{c_1} \parallel \mathbf{c_2})$
10: **return** $\mathbf{c}$

Algorithm 3 `IND-CPA.Decryption`

Input: sk, Ciphertext $\mathbf{c} = (\mathbf{c_1} \parallel \mathbf{c_2})$
Output: Message $\hat{\mathbf{M}}$

1: $\mathbf{c_1}, \mathbf{c_2} \leftarrow$ Parsing$(\mathbf{c})$ and $\mathbf{s} \leftarrow sk$
2: $\hat{\mathbf{M}}' \leftarrow \lfloor (2/q) \cdot ((q/k_2) \cdot \mathbf{c_2} - ((q/k_1) \cdot \mathbf{c_1}) * \mathbf{s}) \rceil$
3: **return** $\hat{\mathbf{M}} \leftarrow \text{eccDEC}(\hat{\mathbf{M}}')$

3.3 Specification of TiGER.CCAKEM

We design IND-CCA KEM using the transformation technique by Jiang et al. [17]. We use a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^{256}$, and a hash function $G : \{0,1\}^* \rightarrow \{0,1\}^*$ for Jiang's transformation technique.

Algorithm 4 `IND-CCA-KEM.KeyGen`

Input: The set of public *parameters*
Output: Public Key $pk = (Seed_a \parallel \mathbf{b})$, Private Key $sk = (sk_{cpa} \parallel \mathbf{u})$

1: $pk, sk_{cpa} :=$ `IND-CPA.KeyGen`$(parameters)$
2: $\mathbf{u} \xleftarrow{\$} R_2$
3: **return** $pk, sk \leftarrow (sk_{cpa} \parallel \mathbf{u})$

Algorithm 5 `IND-CCA-KEM.Encapsulation`

Input: pk
Output: Ciphertext $\mathbf{c} = (\mathbf{c_1} \parallel \mathbf{c_2})$, Shared Key $\mathbf{K}$

1: $\delta \xleftarrow{\$} \{0,1\}^d$
2: $\mathbf{c} :=$ `IND-CPA.Encryption`$(pk,\ \delta\ ;\ H(\delta, H(pk)))$
3: $\mathbf{K} \leftarrow G(H(\mathbf{c}),\ \delta)$
4: **return** $\mathbf{c}, \mathbf{K}$

Algorithm 6 `IND-CCA-KEM.Decapsulation`

Input: pk, sk, Ciphertext $\mathbf{c}$
Output: Shared Key $\mathbf{K}$

1: $\mathbf{s}, \mathbf{u} \leftarrow$ Parsing(sk)
2: $\hat{\delta} :=$ `IND-CPA.Decryption`$(\mathbf{s}, \mathbf{c})$
3: $\hat{\mathbf{c}} :=$ `IND-CPA.Encryption`$(pk,\ \hat{\delta}\ ;\ H(\hat{\delta}, H(pk)))$
4: **if** $\mathbf{c} = \hat{\mathbf{c}}$ **then** $\mathbf{K} \leftarrow G(H(\mathbf{c}), \hat{\delta})$ **else** $\mathbf{K} \leftarrow G(H(\mathbf{c}), \mathbf{u})$
5: **return** $\mathbf{K}$

3.4 Parameter sets

We construct a TiGER128 that satisfies security level 1 (AES128), a TiGER192 that satisfies security level 3 (AES192) and TiGER256 that satisfies security level 5 (AES256) as required by the NIST standardization process. Table 1 shows the detailed parameters of each security level and the bandwidth according to each security level is summarized in Table 2.

n is the dimension of the lattice, q is the modulus of RLWE, p is the modulus of RLWR, k_1 and k_2 are the modulus used for ciphertext compression, h is the hamming weight of the secret key and the ephemeral secret used to encryption. h_{e_1} and h_{e_2} are the hamming weight of the encryption. d is the length of the message, which is related to the security level. f is the number of error bits fixed by error correcting code. Bandwidth is a sum of ciphertext and public key size.

Table 1: The detail parameters for each security level

parameters	security level	n	q	p	k_1	k_2	h_s	h_r	h_e	d	f
TiGER128	AES128	512	256	128	64	16	142	110	32	128	3
TiGER192	AES192	1024	256	128	64	4	132	132	32	256	5
TiGER256	AES256	1024	256	128	128	4	196	196	32	256	5

Table 2: Size of pk, sk, and ciphertext (bytes)

parameters	Ciphertext	Public key	Secret key*
TiGER128	640	480	528
TiGER192	1,024	928	1,056
TiGER256	1,152	928	1,056

* sk_{cpa} can be encoded by storing only non-zero indexes. Thus, optionally, sk can be compressed with $encoding(sk_{cpa})$, a flag of 1 or -1, and $\mathbf{u}$ (for IND-CCA KEM).

3.5 Correctness

The following shows the correctness of our PKE scheme. Let $\hat{\mathbf{M}}$ be an encoded message from eccENC ($\mathbf{M}$), where $\mathbf{M} \in \mathcal{M}$. Let ψ be a positive integer parameter and $\mathbf{s}$, $\mathbf{r}$, $\mathbf{e_1}$, $\mathbf{e_2}$ are randomly chosen from a same distribution $\mathcal{D}$. The value of $\mathbf{e_b}$ is :

$$\mathbf{e_b} = ((\frac{q}{p}) \cdot \lfloor(\frac{p}{q}) \cdot \mathbf{b}\rceil) - \mathbf{b},$$

for some $\mathbf{e_b} \in R_\psi$. The value of $\mathbf{e_{c_1}}$ is :

$$\mathbf{e_{c_1}} = ((\frac{q}{k_1}) \cdot \lfloor(\frac{k_1}{q}) \cdot \mathbf{c_1}\rceil) - \mathbf{c_1},$$

for some $\mathbf{c_1} \in R_\psi$. The value of $\mathbf{e_{c_2}}$ is :

$$\mathbf{e_{c_2}} = ((\frac{q}{k_2}) \cdot \lfloor(\frac{k_2}{q}) \cdot \mathbf{c_2}\rceil) - \mathbf{c_2},$$

for some $\mathbf{c_2} \in R_\psi$. Then, the decryption process is as follows:

$$\begin{aligned}
&\lfloor(\frac{2}{q}) \cdot \big((\frac{q}{k_2}) \cdot \mathbf{c_2} - ((\frac{q}{k_1}) \cdot \mathbf{c_1}) * \mathbf{s}\big)\rceil \\
=&\lfloor(\frac{2}{q}) \cdot \big((\frac{q}{k_2}) \cdot \lfloor(\frac{k_2}{q}) \cdot ((\frac{q}{2}) \cdot \hat{\mathbf{M}} + ((\frac{q}{p}) \cdot \lfloor(\frac{p}{q}) \cdot \mathbf{a} * \mathbf{s}\rceil) * \mathbf{r} + \mathbf{e_2})\rceil \\
&-((\frac{q}{k_1}) \cdot \lfloor(\frac{k_1}{q}) \cdot (\mathbf{a} * \mathbf{r} + \mathbf{e_1})\rceil) * \mathbf{s}\big)\rceil \\
=&\lfloor(\frac{2}{q}) \cdot \big(((\frac{q}{2}) \cdot \hat{\mathbf{M}} + \mathbf{e_b'}\mathbf{r} + \mathbf{e_2'} + \mathbf{e_{c_2}}) - (\mathbf{e_1'}\mathbf{s} + \mathbf{e_{c_1}}\mathbf{s})\big)\rceil \\
=&\hat{\mathbf{M}},
\end{aligned}$$

Let $\mathbf{f} = (\mathbf{e'_b r} + \mathbf{e'_2} + \mathbf{e_{c_2}}) - (\mathbf{e'_1 s} + \mathbf{e_{c_1} s})$, We have that the error rate is $\hat{\epsilon} = 1 - Pr[\lfloor -q/2 \rceil < \mathbf{f} < \lfloor q/2 \rceil]$, since our PKE scheme uses `D2` encode that encodes from one message bit to two coefficients. Using the decryption failure rate estimator of M. Albrecht[2], we obtain that the error rate of each message bit is $2^{-44.28}$. Our PKE scheme uses `XE3` in security level 1 to correct 3-bit errors, we have that

$$\epsilon = 1 - \big(\sum_{f=0}^{3} \binom{512}{f} \cdot ((2^{-44.28})^f) \cdot (1 - 2^{-44.28})^{512-f}\big) \approx 2^{-145.75}.$$

Then, our PKE scheme is (1-ϵ)-correct with $\epsilon < 2^{-128}$, where security parameter λ is 128. The following Table 3 shows the decryption failure rate of our PKE scheme's each parameters. Note that there is a hidden margin in our decryption failure rate calculation. `XE`f accurately corrects errors of f-bits and can correct errors more than f with a high probability. Experimentally `XE5` corrects 99.4% of random 6-bit errors and 97.0% of random 7-bit errors[30].

Table 3: Decryption failure rate

parameters	Bit error rate	Decryption failure rate	f^*
TiGER128	$2^{-44.28}$	$2^{-145.75}$	3
TiGER192	$2^{-33.48}$	$2^{-150.41}$	5
TiGER256	$2^{-41.96}$	$2^{-201.29}$	5

* Let f be the bit length to be corrected by using the XEf.

4 Performance analysis

4.1 Description of platform

We evaluate performance (CPU cycles) used each official code, and the evaluation environment is AMD Ryzen3 2200G @3.5GHz CPU, Ubuntu 22.04.1, GCC 11.3.0 with option $-$O3, and the value is the average for 1,00,000 iterations. Also, our implementation is available to https://github.com/honggoonin/TIGER.git.

4.2 Performance of reference implementation

TiGER is faster 2-2.3 times faster than Kyber reference implementation and 2.6-3.9 times faster than the optimal implementation of LAC (byte size q), with an equivalent security level. This result is an advantage of our scheme's design by choosing q in bytes to the power of 2 and sampling the secret from a sparse ternary uniform distribution. The performance overhead of `XEf` and `D2` is not heavy. A straightforward comparison of the reference implementation with the

[2] https://bitbucket.org/malb/lwe-decryption-failure.git

AVX2 implementation is insignificant, but TiGER is about 30-47% slower than Kyber AVX2. For reference, considering that the first study on AVX2 implementation of LizarMong [19], which has a similar scheme to TiGER, showed a performance improvement of about 21-27%, it can be expected to outperform Kyber if advanced skill. In addition, a recent study showed the application of NTT through modulus switching in the NTT unfriendly parameter where the modulus q is a power of 2, improving Saber by 25-61% and LAC by 2-4 times [12]. Studies like this will have a positive impact on our schemes as well.

Table 4: Performance (CPU cycles)

Algorithm	Key generate	Encapsulation	Decapsulation
TiGER128 (ref)	57,422	73,917	99,121
TiGER192 (ref)	75,131	132,595	159,098
TiGER256 (ref)	91,706	164,539	195,624
Kyber512 (ref[3])	121,721	153,724	189,515
Kyber768 (ref)	217,175	261,818	304,349
Kyber1024 (ref)	308,615	353,579	411,223
LAC128 (opt[4])	138,841	219,415	253,301
LAC192 (opt)	308,557	414,122	638,422
LAC256 (opt)	368,792	595,165	806,561
Kyber512 (AVX2[5])	34,672	47,670	41,675
Kyber768 (AVX2)	59,150	73,523	64,653
Kyber1024 (AVX2)	92,268	121,576	106,296

[3] https://github.com/pq-crystals/kyber.git
[4] https://github.com/pqc-lac/lac-intel64.git
[5] https://github.com/pq-crystals/kyber.git

5 Security

5.1 Security definition

The following is the formal definition of the IND-CPA security.

Definition 3 (IND-CPA). *Let* $\mathsf{PKE} = (\boldsymbol{KeyGen}, \boldsymbol{Encrypt}, \boldsymbol{Decrypt})$ *be a public-key encryption scheme. The security of PKE under chosen plaintext attacks is defined in terms of the following experiment between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*:*

$$\boldsymbol{Exp}_{PKE,\mathcal{A}}^{IND\text{-}CPA}(\lambda)$$

1. $(pk, sk) \leftarrow \boldsymbol{KeyGen}(1^{\lambda})$;
2. $(M_0, M_1) \leftarrow \mathcal{A}$;
3. $\mathcal{C}$ *flips a random coin* $b \in \{0, 1\}$;
4. $ct \leftarrow \boldsymbol{Encrypt}(PK, M_b)$;
5. $b' \leftarrow \mathcal{A}$;

   If $b = b'$, *return 1.*

   Otherwise, return 0.

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{PKE,\mathcal{A}}^{IND\text{-}CPA}(\lambda) = \left|\Pr[b = b'] - \frac{1}{2}\right|$ where the probability is taken over all the randomness of the experiment. A PKE scheme is secure in the security model under chosen plaintext attacks if for all adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above experiment is negligible in the security parameter λ.

5.1.1 Security Assumption

We define decisional Ring Learning With Errors (RLWE) problem and decisional Ring Learning With Rounding (RLWR) problem. Let R_q, R_p denote the rings $\mathbb{Z}_q[x]/(g(x))$, $\mathbb{Z}_p[x]/(g(x))$, where $g(x)$ is an irreducible polynomial of degree n.

Definition 4 (decisional RLWE). *Let* n, q *be positive integers. Let* R_q *be polynomial ring constructed by* $g(x)$*, and let* $\mathfrak{D}_s$ *be a distribution over* R_q*. A decisional RLWE problem* $\mathbf{RLWE}_{n,q}(\mathfrak{D}_s)$ *is to distinguish uniformly random* $(\mathbf{a}, \mathbf{u}) \in R_q \times R_q$ *and* $(\mathbf{a}, \mathbf{b} = \mathbf{a} * \mathbf{s} + \mathbf{e}) \in R_q \times R_q$*, where* $\mathbf{a}$ *is uniform randomly chosen polynomial,* $\mathbf{e}$ *is chosen from error distribution, and* $\mathbf{s}$ *is a secret polynomial. Then, the advantage of an adversary* $\mathcal{A}$ *in solving the decisional RLWE problem* $\mathbf{RLWE}_{n,q}(\mathfrak{D}_s)$ *is defined as follows:*

$$\boldsymbol{Adv}_{n,q}^{RLWE}(\mathcal{A}) = |\mathsf{Pr}[\mathcal{A}(\mathbf{a}, \mathbf{b}) = 1] - \mathsf{Pr}[\mathcal{A}(\mathbf{a}, \mathbf{u}) = 1]|.$$

Definition 5 (decisional RLWR). *Let n, q, p be positive integers such that $q > p$. Let R_q, R_p be polynomial rings constructed by $g(x)$, and let $\mathfrak{D}_s$ be a distribution over R_q. A decisional RLWR problem $\mathbf{RLWR}_{n,q,p}(\mathfrak{D}_s)$ is to distinguish uniformly random $(\mathbf{a}, \mathbf{u}) \in R_q \times R_p$ and $(\mathbf{a}, \mathbf{b} = \lfloor (p/q) \cdot (\mathbf{a} * \mathbf{s}) \rceil) \in R_q \times R_p$, where $\mathbf{a}$ is uniform randomly chosen polynomial, and $\mathbf{s}$ is a secret polynomial. Then. the advantage of an adversary $\mathcal{A}$ in solving the decisional RLWR problem $\mathbf{RLWR}_{n,q,p}(\mathfrak{D}_s)$ is defined as follows:*

$$\boldsymbol{Adv}_{n,q,p}^{RLWR}(\mathcal{A}) = |\mathsf{Pr}[\mathcal{A}(\mathbf{a}, \mathbf{b}) = 1] - \mathsf{Pr}[\mathcal{A}(\mathbf{a}, \mathbf{u}) = 1]|.$$

5.2 Formal Security

5.2.1 Security of IND-CPA PKE We prove that our PKE scheme is IND-CPA secure under the RLWE assumption and the RLWR assumption.

Theorem 1 (IND-CPA PKE). *The above PKE scheme is secure under chosen plaintext attacks if the the RLWE assumption and the RLWR assumption holds. That is, for any PPT adversary $\mathcal{A}$, we have that $\boldsymbol{Adv}_{PKE}^{IND-CPA}(\mathcal{A}) \leq \boldsymbol{Adv}_{n,q}^{RLWE}(\mathcal{B}) + \boldsymbol{Adv}_{n,q,p}^{RLWR}(\mathcal{B})$.*

Proof. The security proof consists of the sequence of hybrid games: The first game will be the original security game and the last one will be a game such that the adversary has no advantage. Let $\hat{\mathbf{M}}$ = eccENC ($\mathbf{M}$). We define the games as follows:

Game G_0 **:** This game is the original security game. In this game, the public key and the ciphertext are properly generated. $\mathcal{D}_0$ is described as follows:

$$\begin{aligned} \mathcal{D}_0 = \{&pk = (Seed_a \parallel \mathbf{b} = \lfloor (\frac{p}{q}) \cdot \mathbf{a} * \mathbf{s} \rceil), \\ &\quad ct = (\lfloor (\frac{k_1}{q}) \cdot (\mathbf{a} * \mathbf{r} + \mathbf{e_1}) \rceil), \lfloor (\frac{k_2}{q}) \cdot ((\frac{q}{2}) \cdot \hat{\mathbf{M}}_b + ((\frac{q}{p}) \cdot \mathbf{b}) * \mathbf{r} + \mathbf{e_2}) \rceil \}. \end{aligned}$$

Game G_1 **:** In the next game, $\mathbf{b}$ in the public key is replaced with uniformly random polynomial in R_p.$\mathcal{D}_1$ is described as follows:

$$\begin{aligned} \mathcal{D}_1 = \{&pk = (Seed_a \parallel \mathbf{b} \xleftarrow{\$} R_p), \\ &\quad ct = (\lfloor (\frac{k_1}{q}) \cdot (\mathbf{a} * \mathbf{r} + \mathbf{e_1}) \rceil), \lfloor (\frac{k_2}{q}) \cdot ((\frac{q}{2}) \cdot \hat{\mathbf{M}}_b + ((\frac{q}{p}) \cdot \mathbf{b}) * \mathbf{r} + \mathbf{e_2}) \rceil \}. \end{aligned}$$

Therefore, $|\Pr[S_0]$ - $\Pr[S_1]| \leq \mathbf{Adv}_{n,q,p}^{RLWR}(\mathcal{B})$.

Game G_2 **:** In the final game, $\mathbf{a} * \mathbf{r} + \mathbf{e_1}$ and $(q/p) \cdot \mathbf{b}) * \mathbf{r} + \mathbf{e_2}$ are replaced with uniformly random polynomial in R_q. Let $\mathbf{u} \xleftarrow{\$} R_q$ and $\mathbf{v} \xleftarrow{\$} R_q$. $\mathcal{D}_2$ is described as follows:

$$\mathcal{D}_2 = \{pk = (Seed_a \parallel \mathbf{b} \xleftarrow{\$} R_p),$$
$$ct = (\lfloor (\frac{k_1}{q}) \cdot \mathbf{u} \rceil), \lfloor (\frac{k_2}{q}) \cdot ((\frac{q}{2}) \cdot \hat{\mathbf{M}}_b + \mathbf{v}) \rceil\}.$$

Therefore, $|\Pr[S_1]$ - $\Pr[S_2]| \leq \mathbf{Adv}^{RLWE}_{n,q}(\mathcal{B})$.

It follows that

$$\begin{aligned}\mathbf{Adv}^{IND-CPA}_{PKE}(\mathcal{A}) &= |Pr[S_0] - Pr[S_2]| \leq |Pr[S_0] - Pr[S_1]| + |Pr[S_1] - Pr[S_2]| \\ &\leq \mathbf{Adv}^{RLWE}_{n,q}(\mathcal{B}) + \mathbf{Adv}^{RLWR}_{n,q,p}(\mathcal{B}),\end{aligned}$$

which concludes the proof of Theorem 1.

5.2.2 Security of IND-CCA KEM We prove that our KEM scheme is IND-CCA secure under the public-key encryption in the quantum random oracle. Using Theorems 1 of Jiang et al. [17], we get the Theorem 2 for the IND-CCA security of our KEM in the quantum random oracle model.

Theorem 2 (IND-CCA KEM in QROM). *We define a public key encryption scheme PKE = (KeyGen, Encrypt, Decrypt) with message space $\mathcal{M}$ and which is (1-ϵ)-correct. For any IND-CCA quantum adversary $\mathcal{A}$ that makes at most q_D queries to the decryption oracle, at most q_G queries to the random oracle G and at most q_H queries to the random oracle H, we have that*

$$\boldsymbol{Adv}^{IND-CCA}_{KEM}(\mathcal{A}) \leq 2q_H \frac{1}{\mathcal{M}} + 4q_G\sqrt{1-\epsilon} + 2(q_G + q_H)\sqrt{\boldsymbol{Adv}^{IND-CPA}_{PKE}(\mathcal{B})}.$$

5.3 Security strength categories

TiGER128, TiGER192, and TiGER256 achieve the security-levels 1, 3, and 5 suggested by the NIST [3], respectively, and the expected security strength for each parameter set is shown in Table 5. The expected security strength is based on MATZOV [24] and was estimated by `LATTICE-ESTIMATOR` [5]. In addition, we considered the combinatorial attack [25].

5.4 Cost of known attacks

We estimate the security strength of RLWR and RLWE, respectively, due to the character of the TiGER scheme, which uses a combination of RLWR for key generation and RLWE for encryption. The RLWR security strength estimate is equivalent to the LWE, whose noise distribution is uniform over the integers in the range $[-q/2p, q/2p)$ [10]. Estimating the security strength of ciphertext additionally considers deterministic noise added due to ciphertext compression in RLWE. For example, if q is 2^8 and k_1 compressing ciphertext $\mathbf{c_1}$ is 2^6, a

Table 5: Computational complexity of best attacks

Parameters		Estimated security strength(log_2)		
		Core-SVP (quantum)	Core-SVP (classical)	MATZOV (classical)
TiGER128	RLWR	119	129	**147**
	RLWE	126	137	149
TiGER192	RLWR	219	231	**246**
	RLWE	231	243	258
TiGER256	RLWR	245	261	**277**
	RLWE	247	263	279

uniform distribution noise (c_{1e}) in the integer range of at least $[-1, 1]$ is added (stdev = 0.82) . If noise e_1 of RLWE generates $HWT(1024, 168)$, the standard deviation is 0.29, so the standard deviation of the final noise is $e_1 + c_{1e} = 0.86$.

We estimate the security strength based on two cost reduction models. First, our estimates are based on the classical and the quantum Core-SVP hardness [6], which is a very conservative underestimation of the real security. Core-SVP is most commonly used in NIST competition, making it easy to compare algorithms. It also allows for a conservative approach in a quantum environment. The conservative estimation of classical Core-SVP is challenging to compare with the number of gates. For example, Kyber512 claims that the classical security strength 2^{118} based on Core-SVP can be converted to a classical gate of $2^{151.5}$ [7], and LightSaber also claims that the computational complexity of 2^{118} is converted into 2^{144} [13]. On the other hand, MATZOV [24] reported that classical gates are stricter than their estimates, Kyber512 has a security strength of $2^{137.5}$ and LightSaber has a security strength of $2^{138.4}$, which does not satisfy the security level. Although a script replicating MATZOV's results has not been released, Albrecht et al. implemented it in `LATTICE-ESTIMATOR` [5], estimating the security strength as 2^{140} for Kyber512 and $2^{137.8}$ for LightSaber, supporting MATZOV's result. Therefore, we present classical cost with Albrecht et al.'s estimator [5] that implements MATZOV, which is the state-of-arts security strength estimation method.

As shown in Table 5, TiGER128, TiGER192, and TiGER256 exceed the classical security strengths of 2^{143}, 2^{207}, and 2^{272} suggested by NIST, respectively. In addition, the classical Core-SVP estimate has a security strength margin of 2^{12}, 2^{18}, and 2^7 compared to Kyber, corresponding to each security level.

In a quantum environment, the impact of MAXDEPTH is a major factor in estimating security strength. NIST has described a plausible value range for MAXDEPTH from 2^{40} logical gates to 2^{64} logical gates [31]. Kyber claim that for the core-SVP-hardness operation estimates to match the quantum gate cost of breaking AES at the respective security levels, a quantum computer would need to support a maximum depth of 2^{70-80} [7]. The TiGER security strength estimates in quantum Core-SVP are slightly insufficient considering the lower limit of the range of MAXDEPTH suggested by NIST, but it has a lot of margin at the upper limit. Also, TiGER has 2^{23}, 2^{28}, and 2^{26} higher than Kyber with the same security level, respectively. Remark, we report the all security strength estimated by the `LATTICE-ESTIMATOR` [5].

As our scheme is based on the RLWE(R) designed by sparse ternary distributions with Hamming weights, it is weak against the combinatorial attack by MAY [25]. By MAY analyzed that time complexity is roughly $S^{0.3}$, and the memory requirement of this attack is roughly $S^{0.25}$, where S is the space of the secret key. Applying May's analysis to our scheme, TiGER128 has a space of secret-key about 2489.78, so time complexity is $2^{146.93}$ and memory is $2^{122.44}$. In the same way, TiGER192 has a time complexity of $2^{208.49}$(a memory requirement of $2^{173.74}$), and TiGER256 has a time complexity of $2^{273.69}$(a memory requirement of $2^{228.08}$).

6 Summary

The PQC-KEM design aims to replace cryptography in legacy security protocols. It would be nice if PQC-KEM had better performance and bandwidth than ECDH or DH for easy migration, but it seems impossible. Therefore, the maximum range of performance and bandwidth acceptable to the security protocol should be carefully considered and determined as the threshold of the scheme design. We decided the bandwidth threshold to be 1,244 bytes based on the IKE-SA-INIT exchange in IKEv2 and the analysis of the Round5 Team[8].

In this paper, we propose TiGER that is an IND-CCA secure KEM based on RLWE(R). We design schemes with ciphertext and public key sizes smaller than 1,244 bytes each to simplify migration to secure protocols. More precisely, TiGER256 with AES256 security level has 1,152 bytes of ciphertext and 928 bytes of public key. A bandwidth smaller than the MTU is advantageous for immigration because it prevents fragmentation in security protocols. Small ciphertext and public key size were achieved by choosing RLWR and RLWE as the base hardness, compressing the ciphertext, and choosing a small modulus $q = 256$. It is also 1.7-2.6 times faster than the reference implementation of Kyber, NIST's first standard KEM. It is the advantage of choosing a sparse ternary uniform distribution. TiGER128, TiGER192, and TiGER256 satisfy security levels 1, 3, and 5, respectively, based on MATZOV, Core-SVP, and meet-attack by May methodology. The high decoding failure rate of the TiGER design achieved a negligible decoding failure rate by correcting errors using `XEf` and `D2` encoding.

In the future, TiGER is expected to achieve good results in the KpqC competition through additional study on the side-channel analysis, optimal implementation, and more efficient attacks.

7 Change Log

2022.12.8. (v1.1) The bit error rate values in Table 3 have been changed. In TiGER (v1.0 for the Kpqc Competition Draft), We entered the Decryption Failure Rate (DFR) without an error-correcting code. However, To obtain the final DFR (with error-correcting code), First, The bit error rate is obtained from the DFR without error-correcting code; Second, the bit error rate was used to calculate the final DFR. The final result(DFR) has not changed, but we acknowledge our mistake. We thank Hyeongmin.Choe, a researcher at Seoul National University, for helpful comments.

2023.2.16. (v2.0) We changed Table 1 because it improved our scheme for D. J. Bernstein's invaluable comment about combinatorial lattice security on the Kpqc bulletin. Table 1 is the parameter set for our scheme. Therefore, Table 2 to Table 5 also change. But, TiGER still satisfied each security level, has negligible decryption failed rate, and has tiny bandwidth and efficient performance.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 99–108 (1996)
2. Akleylek, S., Alkım, E., Tok, Z.Y.: Sparse polynomial multiplication for lattice-based cryptography with small complexity. The Journal of Supercomputing **72**(2), 438–450 (2016)
3. Alagic, G., Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.K., Miller, C., Moody, D., Peralta, R., et al.: Status report on the first round of the NIST post-quantum cryptography standardization process. US Department of Commerce, National Institute of Standards and Technology (2019)
4. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al.: Status report on the third round of the nist post-quantum cryptography standardization process: Nistir 8413. NIST (2022)
5. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)
6. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange—a new hope. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). pp. 327–343 (2016)
7. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber algorithm specifications and supporting documentation. NIST PQC Round **3**(4), 1–43 (2022)
8. Baan, H., Bhattacharya, S., Fluhrer, S.R., Garcia-Morchon, O., Laarhoven, T., Rietman, R., Saarinen, M.J.O., Tolhuizen, L., Zhang, Z.: Round5: Compact and fast post-quantum public-key encryption. IACR Cryptology ePrint Archive **2019**, 90 (2019)
9. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 719–737. Springer (2012)
10. Bogdanov, A., Guo, S., Masny, D., Richelson, S., Rosen, A.: On the hardness of learning with rounding over small modulus. In: Theory of Cryptography Conference. pp. 209–224. Springer (2016)
11. Cheon, J.H., Kim, D., Lee, J., Song, Y.: Lizard: Cut off the tail! practical post-quantum public-key encryption from lwe and lwr. Cryptology ePrint Archive, Report 2016/1126 (2016), `https://eprint.iacr.org/2016/1126`
12. Chung, C.M.M., Hwang, V., Kannwischer, M.J., Seiler, G., Shih, C.J., Yang, B.Y.: Ntt multiplication for ntt-unfriendly rings: New speed records for saber and ntru on cortex-m4 and avx2. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 159–188 (2021)
13. D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In: International Conference on Cryptology in Africa. pp. 282–305. Springer (2018)
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual international cryptology conference. pp. 537–554. Springer (1999)
15. Hoffstein, J., Pipher, J., Silverman, J.H.: Ntru: A ring-based public key cryptosystem. In: International algorithmic number theory symposium. pp. 267–288. Springer (1998)
16. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Theory of Cryptography Conference. pp. 341–371. Springer (2017)

17. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: Ind-cca-secure key encapsulation mechanism in the quantum random oracle model, revisited. In: Annual International Cryptology Conference. pp. 96–125. Springer (2018)
18. Jung, C.G., Lee, J., Ju, Y., Kwon, Y.B., Kim, S.W., Paek, Y.: Lizarmong: Excellent key encapsulation mechanism based on rlwe and rlwr. In: International Conference on Information Security and Cryptology. pp. 208–224. Springer (2019)
19. Kwon, Y.B.: Optimized implementation of lizarmong using avx2. Master's Thesis of Hansung University Graduate School in Rep.Korea pp. 00–00 (2020)
20. Lee, J., Kim, D., Lee, H., Lee, Y., Cheon, J.H.: Rlizard: Post-quantum key encapsulation mechanism for iot devices. IEEE Access **7**, 2080–2091 (2018)
21. Lu, X., Liu, Y., Zhang, Z., Jia, D., Xue, H., He, J., Li, B., Wang, K., Liu, Z., Yang, H.: Lac: Practical ring-lwe based public-key encryption with byte-level modulus. IACR Cryptology ePrint Archive **2018**, 1009 (2018)
22. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 1–23. Springer (2010)
23. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-lwe cryptography. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 35–54. Springer (2013)
24. MATZOV: Report on the Security of LWE: Improved Dual Lattice Attack (Apr 2022). `https://doi.org/10.5281/zenodo.6493704`, `https://doi.org/10.5281/zenodo.6493704`
25. May, A.: How to meet ternary lwe keys. Cryptology ePrint Archive, Paper 2021/216 (2021), `https://eprint.iacr.org/2021/216`, `https://eprint.iacr.org/2021/216`
26. Park, S.H., Kim, S., Lee, D.H., Park, J.H.: Improved ring lwr-based key encapsulation mechanism using cyclotomic trinomials. IEEE Access **8**, 112585–112597 (2020)
27. Peikert, C., Regev, O., Stephens-Davidowitz, N.: Pseudorandomness of ring-lwe for any ring and modulus. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. pp. 461–473. ACM (2017)
28. Peikert, C., Regev, O., Stephens-Davidowitz, N.: Pseudorandomness of ring-lwe for any ring and modulus. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. pp. 461–473 (2017)
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009)
30. Saarinen, M.J.O.: Hila5: On reliability, reconciliation, and error correction for ring-lwe encryption. Cryptology ePrint Archive, Report 2017/424 (2017), `https://eprint.iacr.org/2017/424`
31. of Standards, N.I., Technology: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016)
32. Targhi, E.E., Unruh, D.: Post-quantum security of the fujisaki-okamoto and oaep transforms. In: Theory of Cryptography Conference. pp. 192–216. Springer (2016)